Software Architecture

Lecture 1
Introduction

João Pedro Sousa
George Mason University

outline

- what is software architecture?
  - relation to software engineering
    - relation to engineering
- architectural drivers
- architectural views
- the rest of the course

Acknowledgment
some of the material presented in this course is adapted from 17655, taught to the MSE at CMU by David Garlan and Tony Lattanze

every system has an architecture

- just like every building has an architecture
  - and someone who acted as an architect

- an architecture may be
  - elegant and effective, or
  - clumsy and dysfunctional

examples of software architecture

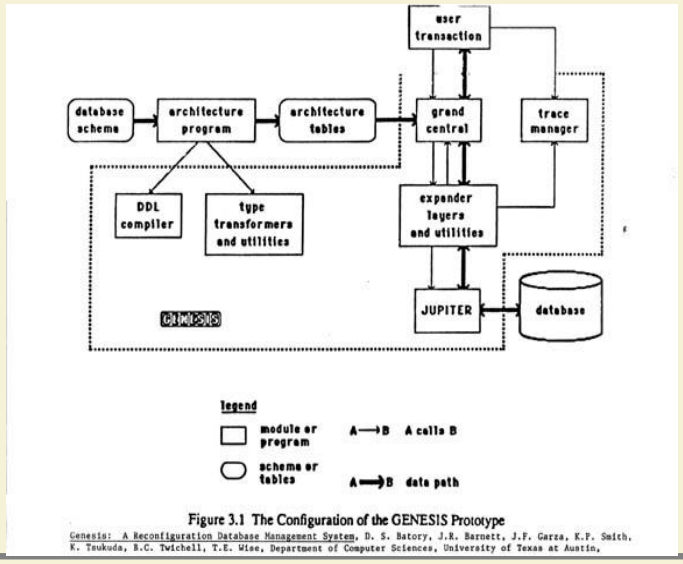


Figure 3.1 The Configuration of the GENESIS Prototype

Genesis: A Reconfiguration Database Management System, D. S. Batory, J.R. Barnett, J.F. Garza, K.P. Smith, K. Tsukuda, B.C. Twichell, T.E. Wise, Department of Computer Sciences, University of Texas at Austin,

more examples

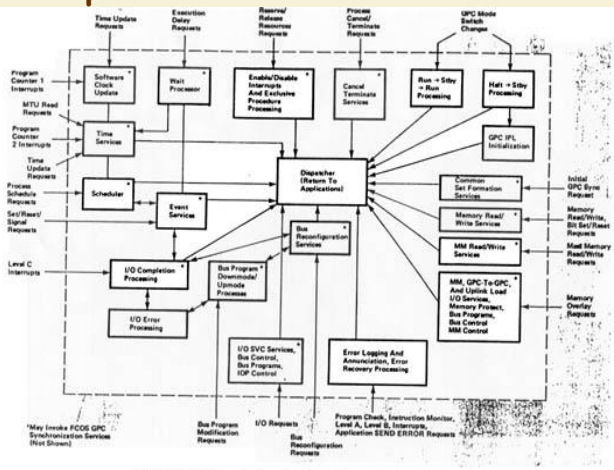


FIGURE 7. Flight Computer Operating System (The FCOS dispatcher coordinates and controls all work performed by the on-board computers.)

Communications of the ACM, "Architecture of the Space Shuttle Primary Avionics Software Systems," Gene D. Carlow, September 1984, Vol. 27, No. 9, P. 933

more examples

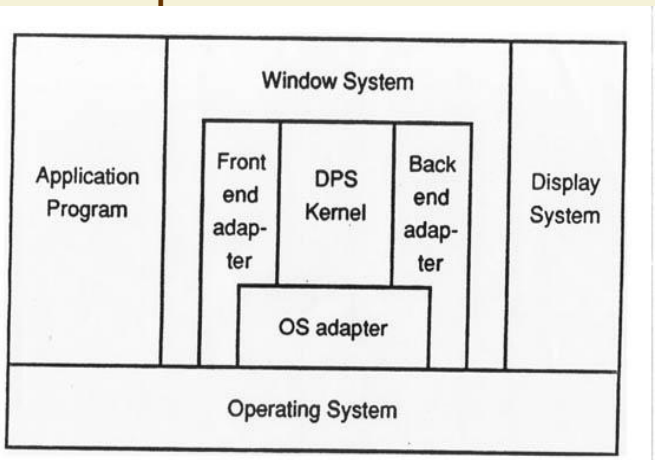


Figure 2. Display PostScript interpreter components.

An Overview of the DISPLAY POSTSCRIPT™ System, Adobe Systems Incorporated, March 16, 1988, P. 10

more examples

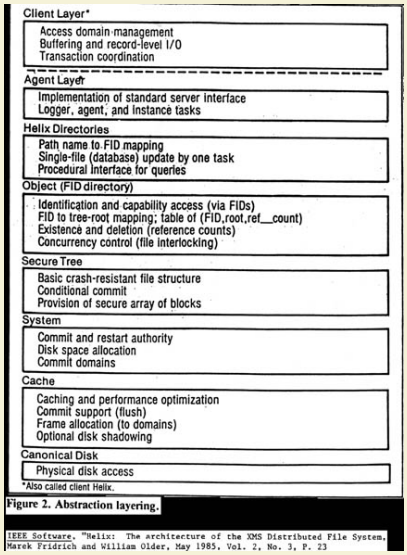


Figure 2. Abstraction layering.

IEEE Software. "Helix: The architecture of the XMS Distributed File System, Marek Fridrich and William Older, May 1985, Vol. 2, No. 3, P. 23

more examples

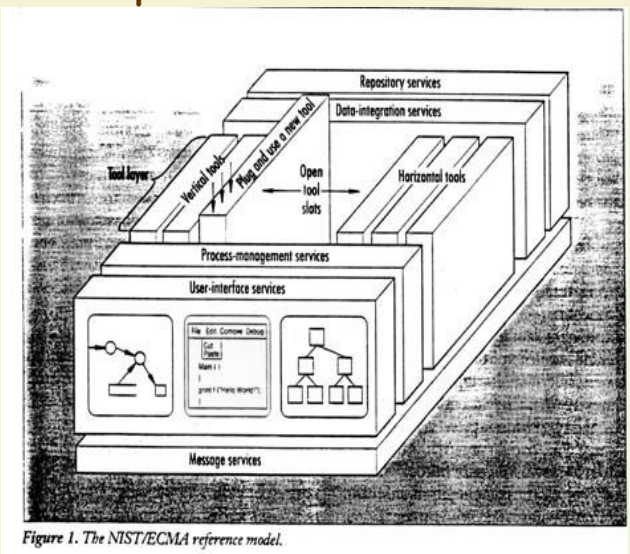
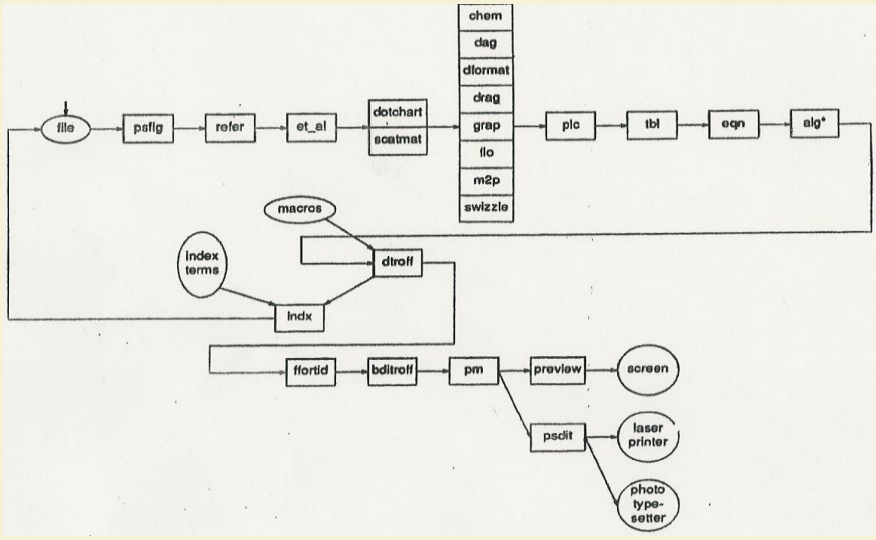


Figure 1. The NIST/ECMA reference model.

more examples



more examples

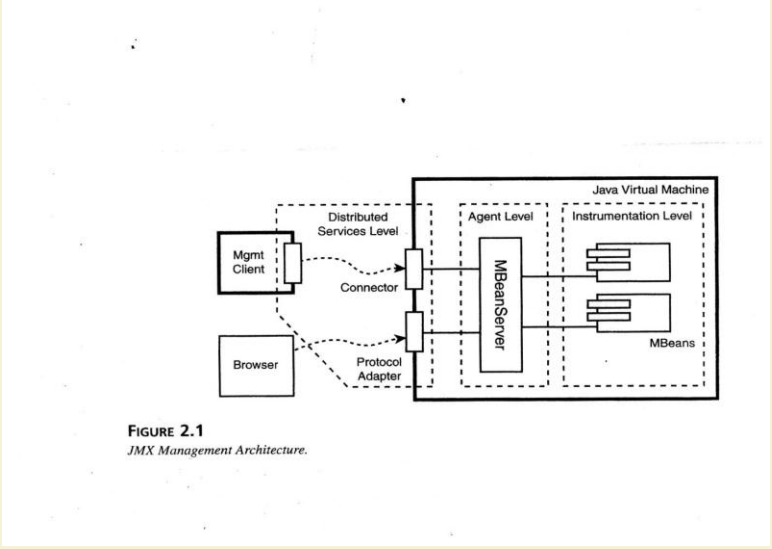
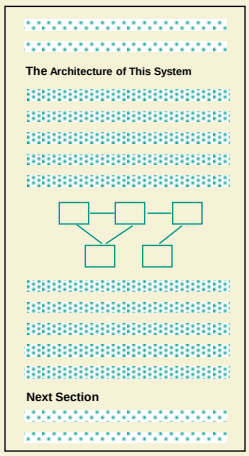


FIGURE 2.1
JMX Management Architecture.

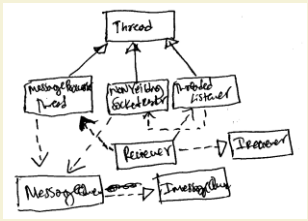
description of a system's architecture

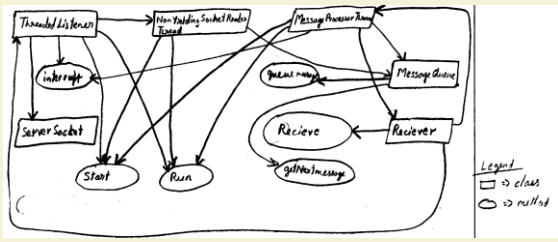


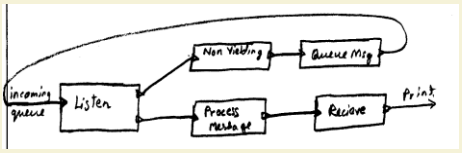
- usually informal prose plus box-and-line diagram
- lots of appeal to intuition
- little precision, rarely formal

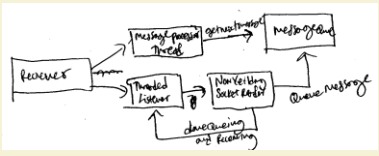
why is it hard?

unlike buildings software is hard to "see"









- architectures produced by different people looking at the same code

good architecture bestows:

- strength
  - durable
  - safe
- beauty
  - balanced proportions
- utility
  - fit planned uses

[Vitruvius, ~40 BCE]

- reliable
- fault-tolerant
- ...
- intelligible
- usable
- maintainable
- ...
- desired features
- quality of service

the challenge

- develop systems "architecturally"
  - apply codified architectural design expertise
  - build systems by assembling existing parts
  - use standards for integration
  - assure that the system has the desired properties
  - reduce development and evolution costs
- turn Software Architecture into an *engineering discipline*
  - from ad hoc definition to codified principles

what is software engineering?

quotes:

- software engineering is, in fact, a form of engineering
  David Parnas (modules, hiding, & OO)
- <it> is not, but it should be
  Steve McConnell (Code Complete, Rapid Development)
- programming is an art and a science
  Donald Knuth (computational complexity, analysis of algorithms)

what is engineering?

definitions abound. They have in common:

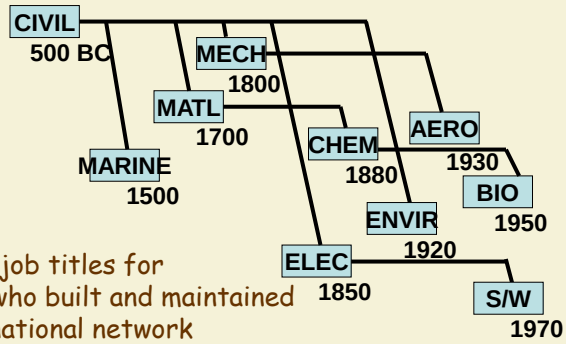
Creating cost-effective solutions ...
... to practical problems ...
... by applying scientific knowledge ...
... building things ...
... in the service of mankind

engineering enables ordinary people
to do things that formerly required virtuosos

engineering entails making decisions under the
constraints of limited time, knowledge, and resources

what is engineering?

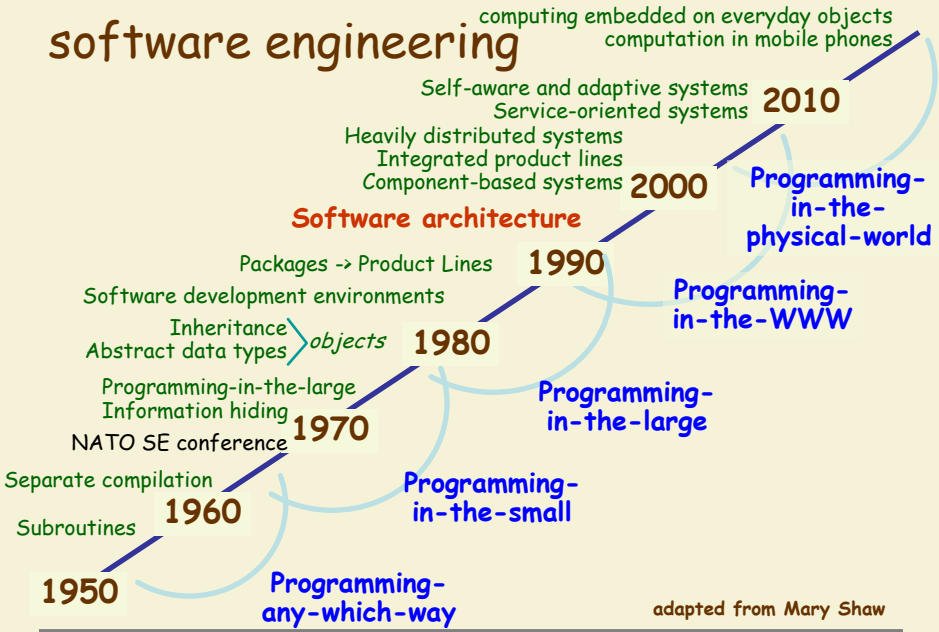
- before the 19th century the kinds of *systems* that engineers studied and built had little technological or mechanical aspects



- by 1920, AT&T had job titles for *System Engineers*, who built and maintained *The System*: their national network

Honour, E. "Characteristics of Engineering Disciplines"
13th Int'l Conference on Systems Engineering, 1999

software engineering



adapted from Mary Shaw

many angles
into software engineering

- managing software production
  - who does what, when – lifecycle/process/risk management
  - methods, tools, skills
- best practices/tools of the trade
  - systematic testing/peer reviews
  - separation of concerns/aspects
  - rapid prototyping/configuration management/evolution
  - product lines/design patterns
- correlating structure and properties
  - software economics: size/complexity <-> cost
  - **Software Architecture**:
    design patterns <-> *-ility tradeoffs

software architect:
a recognized trade within SE

human-computer interaction

applications

development/execution envs.

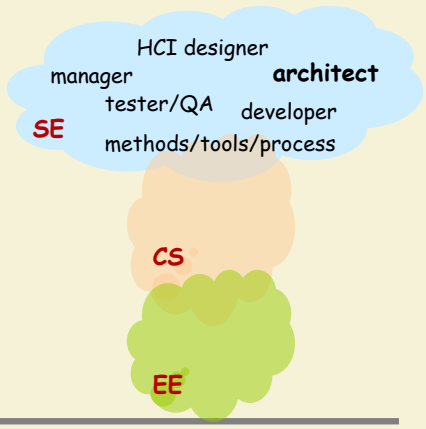
middleware

OS/storage management

networking

hardware

interfaces with:
psychology, ergonomics, human factors



research questions
areas addressed by SA

- To-find - *is there an X, and what is it?*
- To-show - *is X always true of Y ?*
- Feasibility - *is it possible to accomplish X at all?*
- Method - *how do I accomplish X?*
- Means - *what mechanism will do X? how can I automate X?*
- **Characterization** - *what are important characteristics of X? what's X like? what, exactly, do we mean by X ?*
- **Classification** - *what are the varieties of X? and how are they related?*
- **Prediction** - *given X, what will Y be?*
- **Discrimination** - *how do I decide whether X or Y?*

what is software architecture?

Definitions abound. They have in common:

the structure
or structures of the system
which comprise elements
their externally-visible properties
and the relationships among them

SA & programming address different issues

Architecture	*Programs*
interactions among parts	implementations of parts
system-wide views	local views
declarative	operational
more stable components	dynamic allocation
system-level performance	algorithmic performance

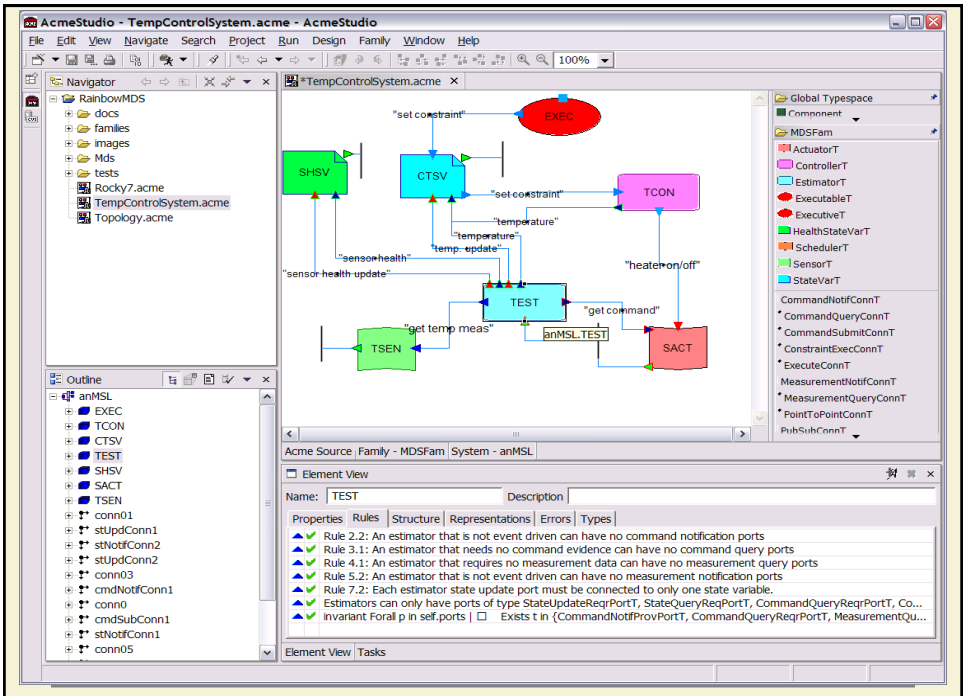
- software engineering is, in fact, a form of engineering
  David Parnas
- programming is an art and a science
  Donald Knuth

SA makes a difference

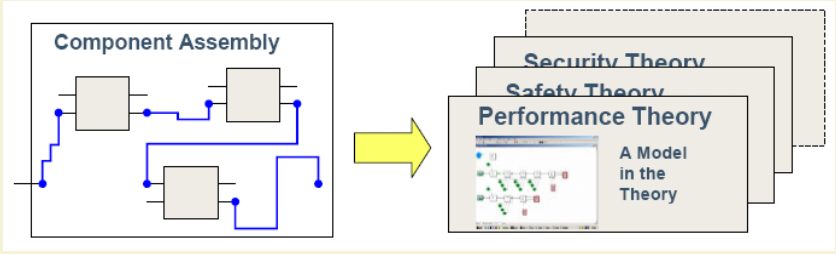
- design Reviews at ATT
  - Architecture Review Board created 1988
  - facilitates architectural reviews for projects
- results
  - reviewed over 350 projects from 1989-95
  - "a correct architecture has the largest single impact on cost and quality of the product" (Maranzano 1995)
  - "based on experience … we found an average savings of at least one-half staff year per project reviewed, and substantially larger savings on several projects reviewed."
  - estimated cumulative savings: approx 175 staff years

SA in the 2000's

- incorporation of architectural notions into mainstream *design languages* (e.g., UML-2), and *tools* (e.g., Rose)
- *methods* based on architectural design and refinement (e.g., Model-Driven Architecture – MDA)
- some architecture *analysis tools*
- architectural *standards* for enterprise systems (e.g., RM-ODP, TOGAF)



SA correlates design patterns/styles with quality attribute **tradeoffs**



- using formal/quantitative models of systems and environment

take 5

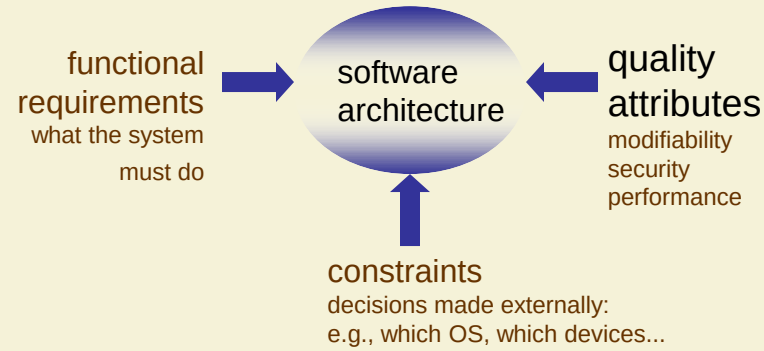
outline

- what is software architecture?
- architectural drivers
- architectural views
- the rest of the course

structure matters

- two systems may have different structure and provide the same function
- what are the effects of architectural decisions on quality attributes?
  - each structure promotes different qualities
    - design for modifiability
    - design for security
    - design for performance...
- many architectural decisions concern QAs

architectural drivers
requirements that shape the architecture



are QAs
non-functional requirements?

- yes, but:
  the term *non-functional* suggests a false partitioning
- "ility" names are ***not*** enough
  - vocabulary varies widely
    there is no widely accepted standard
  - descriptions are vague and lack quantifiable measures
  - dictionary definitions are superficial
  - debates on what "ilities" really mean are not productive
- QAs cannot be described independently of functionality

much more on this later in the course

focus on **run-time** structure & **quality attributes**

- performance
- security
- availability
- recoverability
- safety
- scalability
- ...
- usability
  - learnability
  - customizability
  - adequacy
  - ...
- maintainability
  - extensibility
  - testability
  - interoperability
  - portability
  - ...

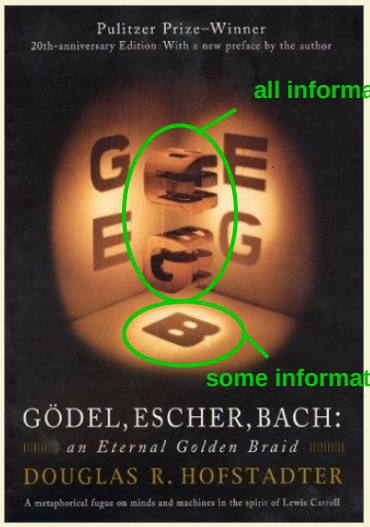
outline

- what is software architecture?
- architectural drivers
- architectural views
  - module viewtype
  - component & connector viewtype
  - allocation viewtype
  - styles
- the rest of the course

one system, many views

- a view is a representation of a set of system elements and the relations among them
- not *all* system elements
- a view selects *element types* and *relation types* of interest, and shows only those

why?



one system, many views

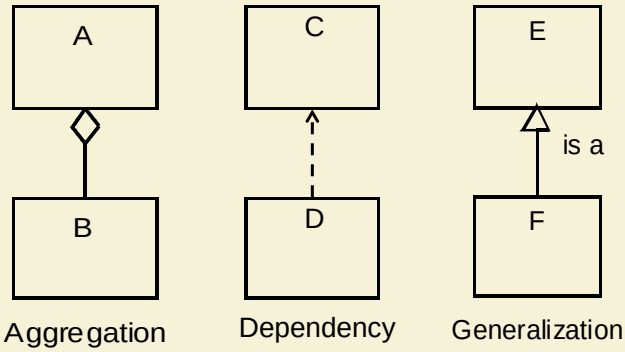
- an architect examines the system in three ways
  - how is it structured as a set of code units?
    - module viewtype
  - how is it structured as a set of elements that have run-time behavior and interactions?
    - component & connector viewtype
  - how does it relate to non-software structures, such as hardware and development teams?
    - allocation viewtype

module viewtype
describes the code structure

- elements are modules
  code unit that implements a set of functionalities
- relations among modules include
  - A is part of B
    defines a part-whole relation
  - A depends on B
    defines a functional dependency relation
  - A is a B
    defines specialization and generalization

different notations exist
for module views

- UML class diagrams:



- informal: stacked boxes, box-and-line...

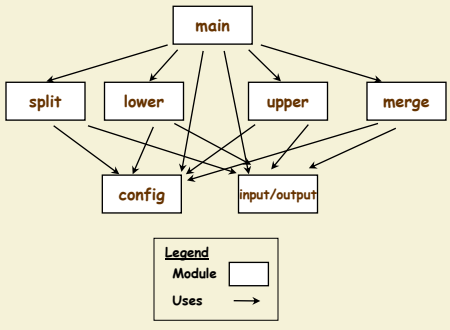
C&C viewtype
describes how the system works

- elements
  - components (boxes)
    principal units of run-time computation and data stores
  - connectors (lines)
    interaction mechanisms – identity and behavior of their own
- relations
  - attachment of components to connectors
- properties
  information for construction & analysis
  - quality attributes (more in a moment)

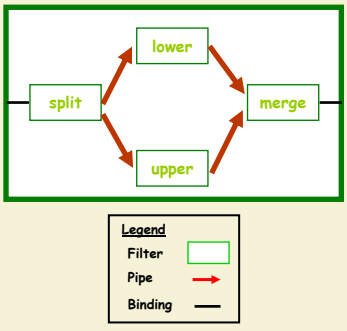
module and C&C show different aspects

example program:
- produce alternating case of characters in a stream

module view



C&C view



C&C viewtype
used for behavior and QoS analysis

- construction
  - how the system will appear at run time
  - what kind of behavior must be built in
  - pathways of interaction and communication mechanisms
- analysis of runtime properties, aka QoS
  - availability
  - performance
  - security
  - reliability…

architectural styles:
specialization of element and relation types

- within each viewtype, recurring forms have been widely observed in different systems
- these forms, or patterns, are worth capturing because they have known properties and can be re-used:
  "tools" in the architect's "bag of tricks"

an architectural style
is a specialization of element and relation types
together with a set of constraints on how they can be used

- styles exist independently of any system
- two different systems can use the same style
- different parts of the same system may use different styles

many styles
in the C&C viewtype

data flow
- batch sequential
- dataflow network (pipe & filter)
  - acyclic, fan-out, pipeline, Unix
- closed loop control

call-return
- main program/subroutines
- information hiding
  - objects, naive client-server

interacting processes
- communicating processes
  - LW processes, distributed objects
- event systems
  - implicit invocation
  - publish-subscribe

data-oriented repository
- transactional databases
  - true client-server
- blackboard
- modern compiler

data-sharing
- compound documents
- hypertext
- Fortran COMMON
- LW processes

hierarchical
- tiers
  - interpreter
  - N-tiered client-server

in Summary

- views help manage the complexity of describing an architecture
- viewtypes
  determine the kinds of things a view talks about
  - three primary viewtypes: module, C&C, allocation
- each viewtype has many styles
  - module: decomposition, generalization, layered, …
  - C&C: pipe & filter, client-server, pub-sub…
  - allocation: deployment, work assignment…

in Summary

viewtypes help analyze system properties

- module: code properties
  - maintainability: extensibility, portability …
- allocation: engineering properties
  - buildability, testability, …
- C&C: runtime properties, aka QoS
  - performance , security, recoverability, …

SA in the 2010's

http://money.cnn.com/magazines/moneymag/bestjobs/2010/

outline

- what is software architecture?
- architectural drivers
- architectural views
- the rest of the course